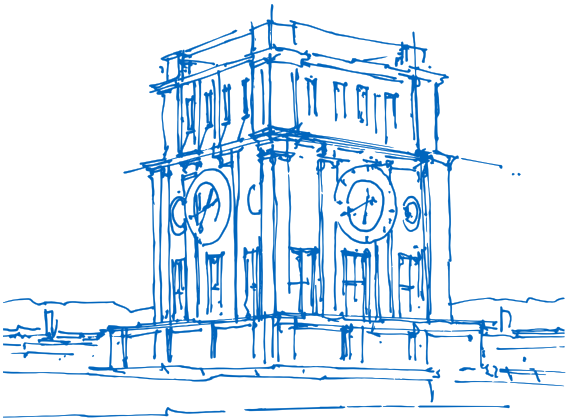
School of Computation, Information and Technology
Technische Universität München

Übung 08: RISC-V Single-Cycle Prozessor

Einführung in die Rechnerarchitektur



Michael Morandell

School of Computation, Information and Technology
Technische Universität München

9. – 15. Dezember 2024

Mitschriften & Infos

Montags:
https://zulip.in.tum.de/#narrow/stream/2668-ERA-Tutorium—Mo-1000-4

Donnerstags:
https://zulip.in.tum.de/#narrow/stream/2657-ERA-Tutorium—Do-1200-2

Website: https://home.in.tum.de/ momi/era/

Keine Garantie für die Richtigkeit der Tutorfolien.

Bei Unklarheiten/Unstimmigkeiten haben VL/ZÜ-Folien recht!

Inhaltsübersicht

- Quiz
- Wiederholung
- Tutorblatt
  - Maschinensprache
  - Fehlerhafte Kontrollsignale
  - Erweiterung: BNE
  - Kontrollsignalbelegung

Zitat der Woche

„So funktioniert die Welt am Ende auch. Man braucht halt seine Vollidioten für jedes Gebiet – ich bin der Vollidiot für Informatik. Es gibt andere Vollidioten für Elektrotechnik. So halten wir die Welt zusammen.“

– Prof. Dr. Robert Wille (Vollidiot für Informatik)

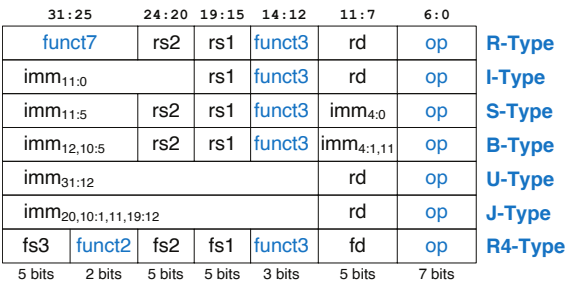
Quelle: Lecture: November 26. 2024 (tum.live)

RISC-V Instruktionstypen

- R-Typ: Register-Register-Operationen (bspw. `add`, `sub`, `sll`)
- I-Typ: kleine Immediates (12 Bit) und Ladebefehle (bspw. `jalr`, `lw`, `ori`)
- U-Typ: große Immediates (20 Bit) (bspw. `lui`, `auipc`)
- S-Typ: Speicherbefehle (bspw. `sw`, `sh`)
- B-Typ: Branches (bedingte Sprünge) (bspw. `beq`, `blt`, `bgtu`)
- J-Typ: Jumps (unbedingte Sprüge) (`jal`)

Assemblierung zu Maschinensprache

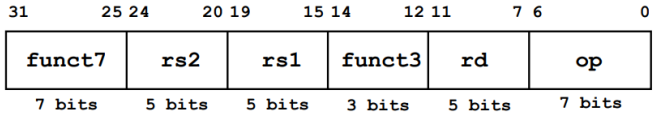
- RV32: Instruktionsgröße von 32 Bit
- Opcode (7-Bit) spezifiziert Instruktionstyp
- Instruktionen selben Typs haben gleiches Instruktionsformat
- funct3 (3-Bit) & funct7 (7-Bit) spezifizieren konkrete Funktionalität innerhalb des Opcodes (z.B. add, sub usw. im R-Typ)



(Quelle: Vorlesungsmaterialien ERA)

Instruktionsformat R-Typ

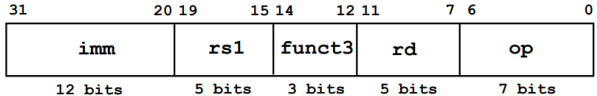
- Instruktionsformat R-Typ (Register-Format) wird meist für arithmetische und logische Instruktionen verwendet



- Felder
  - `funct7` Funktionscode 7 Bit (function) *
  - `rs2` Register des zweiten Quelloperanden (source 2)
  - `rs1` Register des ersten Quelloperanden (source 1)
  - `funct3` Funktionscode 3 Bit (function) *
  - `rd` Register, in dem das Ergebnis gespeichert wird (destination)
  - `op` Operationscode (OP-Code) * : 51 (arithm. und logische)

Instruktionsformat I-Typ

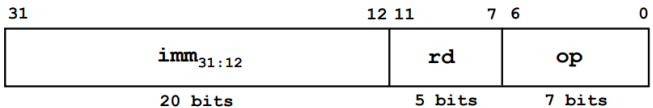
- Instruktionsformat I-Typ (Immediate-Format) wird verwendet für
  - Immediate-Versionen der arithmetischen und logischen Instruktionen, Datentransferinstruktionen (ladend) und für unbedingte Sprünge



- Felder
  - `immediate` 12-Bit Konstante Sign Extended (Werte von -2^{11} bis $2^{11} - 1$)
  - `rs1` Quellregister
  - `funct3` Funktionscode 3 Bit (function)
  - `rd` Zielregister
  - `op` Operationscode (OP-Code): 3 (load), 19 (arithm. und logisch), 103 (Sprünge)

Instruktionsformat U-Typ

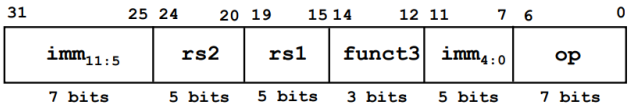
- Instruktionsformat U-Typ (Upper Immediate-Format) wird verwendet für
  - Instruktionen um die höheren 20-Bit in Registern mit Immediate Werten ansprechen zu können



- Felder
  - `immediate` 20-Bit Konstante
  - `rd` Zielregister
  - `op` Operationscode (OP-Code): 23, 55

Instruktionsformat S-Typ

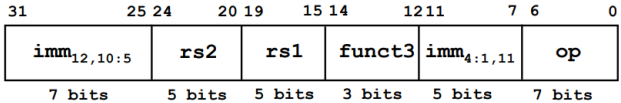
- Instruktionsformat S-Typ (Store-Format) wird verwendet für
  - Datentransferinstruktionen (speichernd)



- Felder
  - `immediate` 12-Bit Konstante aufgeteilt auf 2 Felder (Werte von -2^{11} bis $2^{11}-1$)
  - `rs1` Basisregister
  - `rs2` Wert, der gespeichert werden soll
  - `funct3` Funktionscode 3 Bit (function)
  - `op` Operationscode (OP-Code): 35

Instruktionsformat B-Typ

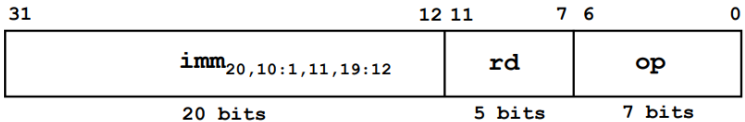
- Instruktionsformat B-Typ (Branch-Format)
  - Wird verwendet für Bedingte Sprünge



- Felder
  - `immediate` 13-Bit Konstante aufgeteilt auf 2 Felder (Werte von -2^{11} bis $2^{11}-1$)
  - `rs1, rs2` Registeroperanden (Vergleichsregister)
  - `funct3` Funktionscode 3 Bit (function)
  - `op` Operationscode (OP-Code): 99

Instruktionsformat J-Typ

- Instruktionsformat J-Typ (Jump-Format)
  - wird für unbedingte Sprünge verwendet

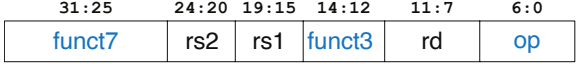


- Felder
  - `immediate` 20-Bit Konstante
  - `rd` Zielregister
  - `op` Operationscode (OP-Code): 111

Beispiel: Assemblierung zu Maschinensprache

```
xor t2, t1, t0
```

op	funct3	funct7	Type	Instruction
0110011 (51)	100	0000000	R	`xor rd, rs1, rs2`



R-Type

1. `xor` → R-Typ
2. `t0` → `x5` (rs2), `t1` → `x6` (rs1), `t2` → `x7` (rd)
3. funct7, funct3, op aus Tabelle ablesen

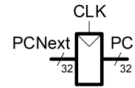
$$(0000000\ 00101\ 00110\ 100\ 00111\ 0110011)_2 = 0\text{x}005343\text{B}3$$

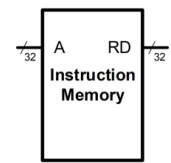
Prozessor-Komponenten

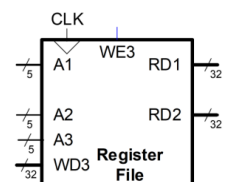
- **Program Counter (PC)**: Adresse des nächsten Befehls

- **Instruktionsspeicher**
  - A: Adresse
  - RD (Read Data): Gelesene Instruktion
  - Zunächst starke Vereinfachung: nur ROM, Lesen damit rein kombinatorisch

- **Registerbank**
  - 2 Ports zum Lesen von Operanden (RD1, RD2; zugehörige Adressen A1 und A2)
  - 1 Port zum Schreiben des Resultats (WD3 und Adresse A3)
  - Es wird nur geschrieben wenn WE3=1 ist (Write Enable)

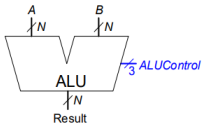




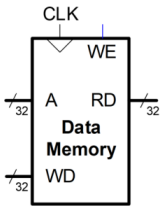


Prozessor-Komponenten

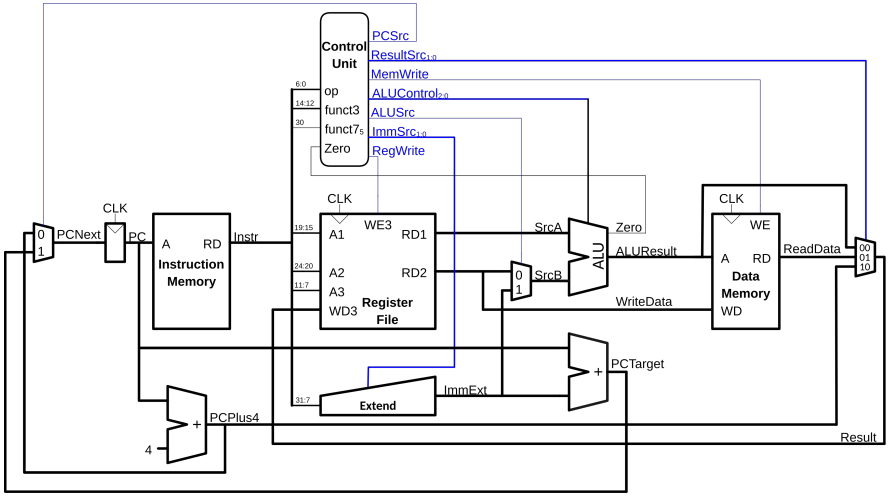
- **Arithmetic Logic Unit (ALU)**
  - Rein kombinatorisch
  - 32-Bit Ausgang mit Ergebnis
  - 1-Bit Ausgang, der angibt, ob Ergebnis 0 ist (nicht dargestellt)



- **Datenspeicher**
  - A: Adresse zum Lesen/Schreiben (Wortadressierung)
  - RD (Read Data): Gelesenes Datum (lesen ist rein kombinatorisch)
  - WD (Write Data): Zu schreibendes Datenwort
  - Geschrieben wird nur bei positiver Taktflanke und wenn WE=1



RISC-V Single-Cycle-Prozessor



(Quelle: Vorlesungsmaterialien ERA)

Feedback

https://tinyurl.com/era-tut

Ein Teil der Folien stammt aus dem Foliensatz von Niklas Ladurner. Vielen Dank dafür!